MINI PROJECT PRESENTATION SLR + SLC + USL

GROUP 3

Team Members: Aditya Aryan | Bhavani Kanaparthi | Naman Goswami | Neha Mondal

Introduction to the Problem Statement

A. New Bengaluru restaurants struggle to determine the cost per two customers for a single dining experience. As part of Zomato's analyst team, we are tasked with providing insights into the reasons customers consider when choosing a restaurant and developing a predictive model to estimate the cost for two people. Our goal is to support these restaurants by equipping them with the knowledge they need to thrive in the industry.

B. Zomato faces challenges in attracting customers with diverse items and offers due to a decline in offline orders, leading to a decrease in user subscriptions. To address this issue, Zomato has entrusted the project to us, aiming to determine whether customers would prefer ordering online or offline. The problem statement revolves around classifying orders as either online or offline and uncovering the underlying patterns that drive these choices.

Data Description

- Order details - URL, address and name of the restaurant, type of order, table booking etc.
- 51717 entries
- 17 attributes
- 16 categorical attributes
- 1 numerical attribute
- 19720 duplicates
- 19.44 maximum null percentage

	url	address	name	online_order	book_table	rate	votes	phone
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	080 42297555\r\n+91 9743772233
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	080 41714161
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918	+91 9663487993

Data Cleaning Steps

1. Converted rate and approx. cost to numeric by cleaning
2. Dropped redundant and high cardinality columns
3. Dropped <1% nulls
4. Imputed remaining
5. Dropped duplicates
6. Capped outliers of 'rate'

```
# cleaning of 'rate'

# function replaces all 'NEW' and '-' values with null so that we can handle them easily
# function also removes the '/' part to change the value into float
def clean_rate(val):
    if val == "NEW" or val == "-":
        return np.nan
    else:
        val = str(val).split('/')
        val = val[0] # takes only the part before '/'
        return float(val)
d.rate = d.rate.apply(clean_rate)
d.rate.unique()
```

```
array([4.1, 3.8, 3.7, 3.6, 4.6, 4. , 4.2, 3.9, 3.1, 3. , 3.2, 3.3, 2.8,
       4.4, 4.3, nan, 2.9, 3.5, 2.6, 3.4, 4.5, 2.5, 2.7, 4.7, 2.4, 2.2,
       2.3, 4.8, 4.9, 2.1, 2. , 1.8])
```

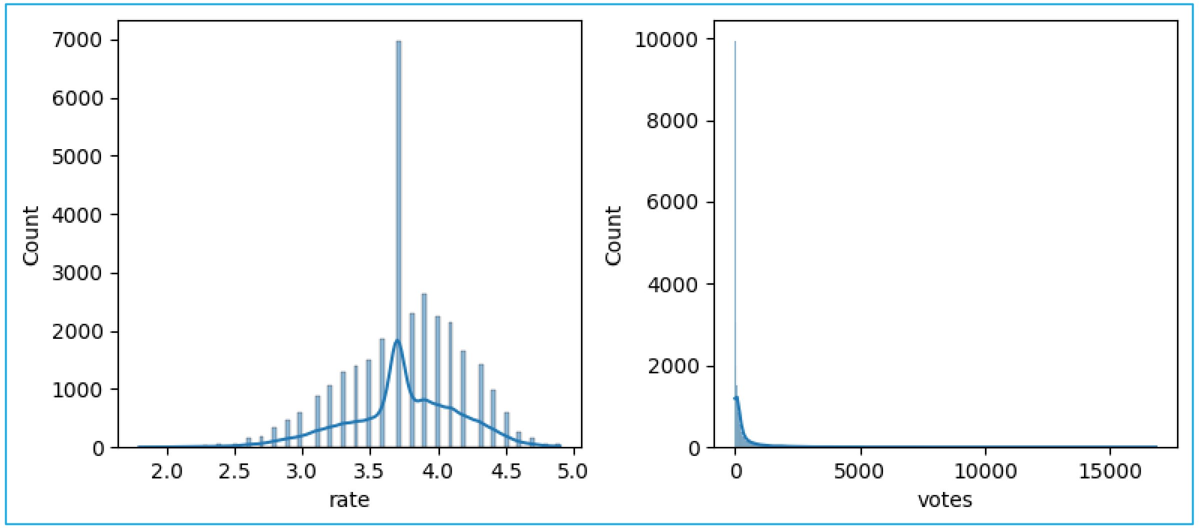
```
# cleaning of 'approx_cost(for two people)'

def clean_cost(value):
    value = str(value)
    value = value.replace(',','') # removes ','
    return float(value)
d['approx_cost(for two people)'] = d['approx_cost(for two people)'].apply(clean_cost)
d['approx_cost(for two people)'].unique()
```

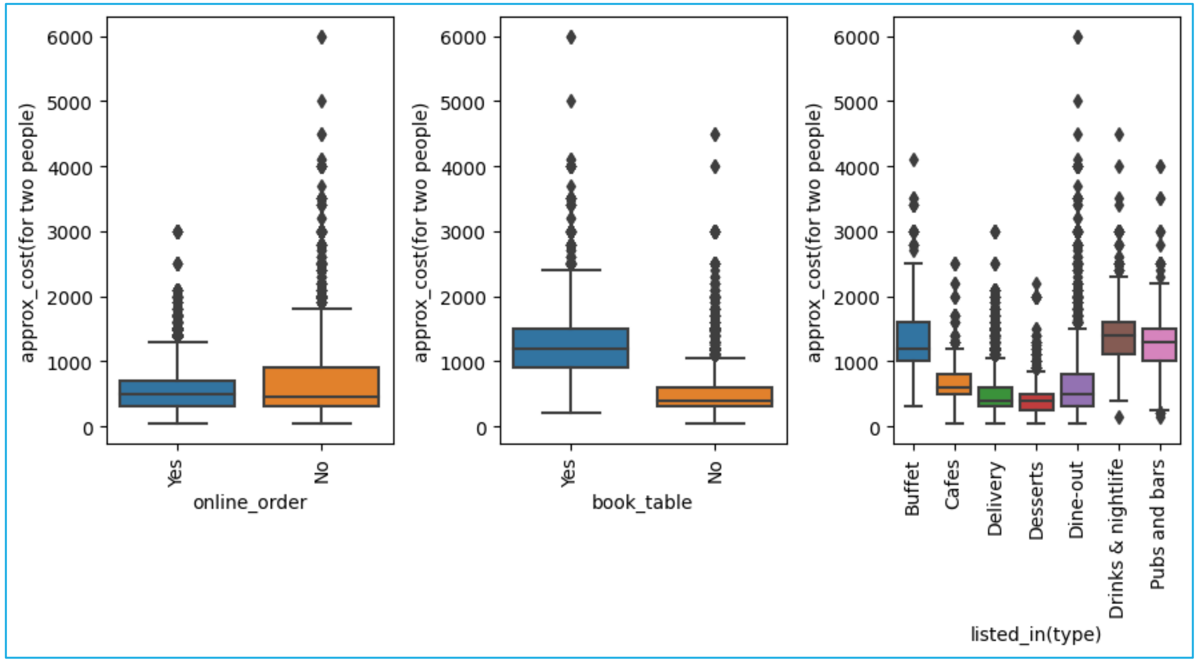
```
array([ 800.,  300.,  600.,  700.,  550.,  500.,  450.,  650.,  400.,
        900.,  200.,  750.,  150.,  850.,  100., 1200.,  350.,  250.,
        950., 1000., 1500., 1300.,  199.,   80., 1100.,  160., 1600.,
        230.,  130.,   50.,  190., 1700.,   nan, 1400.,  180., 1350.,
       2200., 2000., 1800., 1900.,  330., 2500., 2100., 3000., 2800.,
       3400.,   40., 1250., 3500., 4000., 2400., 2600.,  120., 1450.,
        469.,   70., 3200.,   60.,  560.,  240.,  360., 6000., 1050.,
       2300., 4100., 5000., 3700., 1650., 2700., 4500.,  140.])
```

Problem Solving Steps

Section A

- Checked datatypes
- Analyzed summary stats
- Univariate and bivariate analysis
- Tested effect of features on target
- Encoding
- Split data into train-test
- Fit base model
- Fit final model





```python
d.name=LabelEncoder().fit_transform(d.name)
d.online_order=d.online_order.map({'Yes':1,'No':0}) # replaces 'Yes' with 1 and 'No' with 0
d.book_table=d.book_table.map({'Yes':1,'No':0}) # replaces 'Yes' with 1 and 'No' with 0
d.location=d.location.map(dict(d.groupby('location')['approx_cost(for two people)'].mean()))
# performs target encoding
d.rest_type=LabelEncoder().fit_transform(d.rest_type)
d['listed_in(type)']=d['listed_in(type)'].map(dict(d.groupby('listed_in(type)')
                                                  ['approx_cost(for two people)'].mean()))
# performs target encoding
d.head()
```

	name	online_order	book_table	rate	votes	location	rest_type	approx_cost(for two people)	listed_in(type)
0	3664	1	1	4.1	775	426.125000	27	800.0	1338.467742
1	6969	1	0	4.1	787	426.125000	27	800.0	1338.467742
2	6450	1	0	3.8	918	426.125000	22	800.0	1338.467742
3	198	0	0	3.7	88	426.125000	78	300.0	1338.467742
4	2919	0	0	3.8	166	346.759907	27	600.0	1338.467742

OLS Regression Results

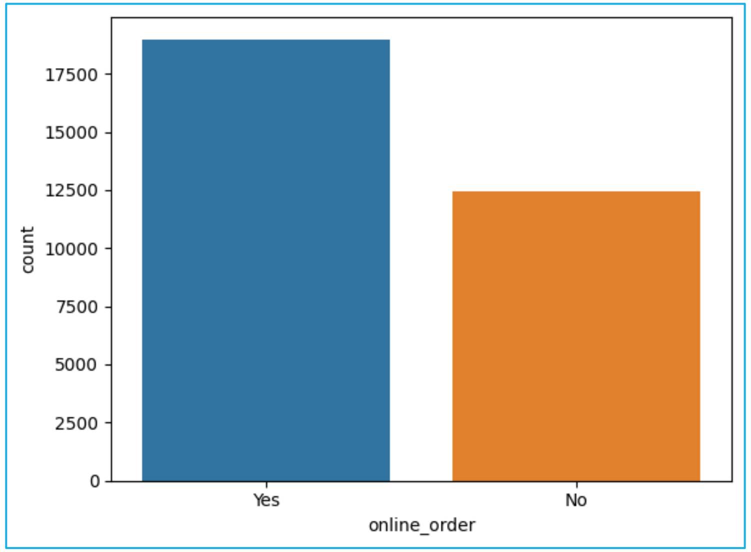
Dep. Variable:	approx_cost(for two people)	**R-squared:**	0.560
Model:	OLS	**Adj. R-squared:**	0.560
Method:	Least Squares	**F-statistic:**	3499.
Date:	Wed, 10 May 2023	**Prob (F-statistic):**	0.00
Time:	16:55:35	**Log-Likelihood:**	-1.5766e+05
No. Observations:	22012	**AIC:**	3.153e+05
Df Residuals:	22003	**BIC:**	3.154e+05
Df Model:	8		

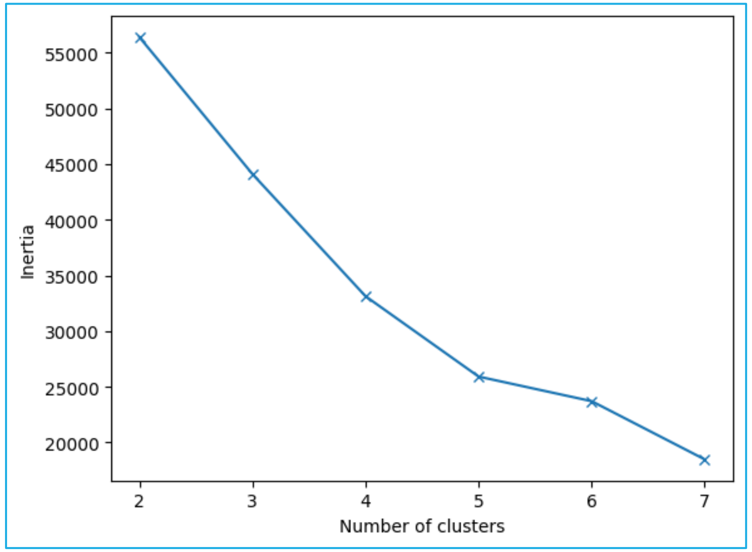
Omnibus:	13443.165	**Durbin-Watson:**	1.989
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	306105.588
Skew:	2.530	**Prob(JB):**	0.00
Kurtosis:	20.554	**Cond. No.**	6.13e+04

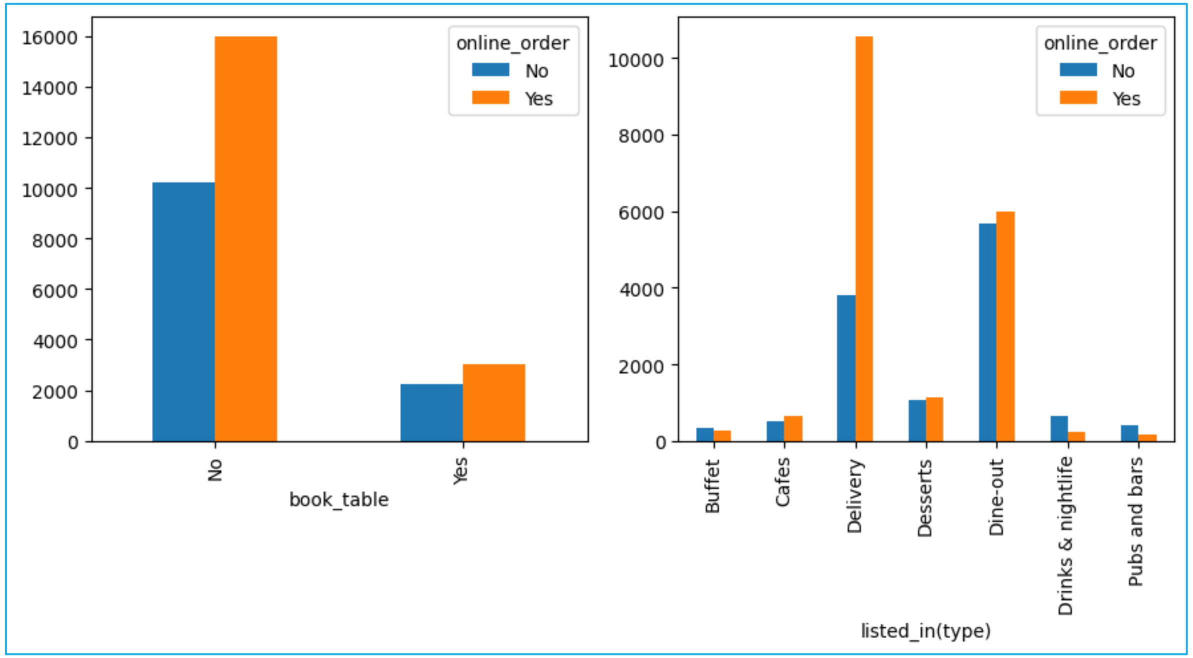
	coef	std err	t	P>|t|
const	-217.6451	24.771	-8.786	0.000
name	0.0038	0.001	4.574	0.000
online_order	-43.3750	4.415	-9.825	0.000
book_table	487.3829	6.945	70.182	0.000
rate	81.0170	6.286	12.889	0.000
votes	0.0371	0.003	14.840	0.000
location	0.4670	0.011	43.465	0.000
rest_type	-2.6366	0.081	-32.608	0.000
listed_in(type)	0.4660	0.011	43.695	0.000

Section B

- Analysed target variable
- Bivariate analysis
- Scaled numeric data
- Created elbow plot
- Fit K-means with optimal K
- Interpreted clusters
- Tested effect of features on target
- Encoding
- Fit base model
- Tried different models
- Fit final model







```
d.name=LabelEncoder().fit_transform(d.name)
d.online_order=d.online_order.map({'Yes':1,'No':0}) # replaces 'Yes' with 1 and 'No' with 0
d.book_table=d.book_table.map({'Yes':1,'No':0}) # replaces 'Yes' with 1 and 'No' with 0
d.location=LabelEncoder().fit_transform(d.location)
d.rest_type=LabelEncoder().fit_transform(d.rest_type)
d['listed_in(type)']=d['listed_in(type)'].map(dict(d.groupby('listed_in(type)').online_order.mean()))
# performs target encoding
d.head()
```

	name	online_order	book_table	rate	votes	location	rest_type	approx_cost(for two people)	listed_in(type)	cluster
0	3664	1	1	4.1	775	1	27	800.0	0.432258	1
1	6969	1	0	4.1	787	1	27	800.0	0.432258	1
2	6450	1	0	3.8	918	1	22	800.0	0.432258	1
3	198	0	0	3.7	88	1	78	300.0	0.432258	0
4	2919	0	0	3.8	166	4	27	600.0	0.432258	0

Base Model

```
lr=LogisticRegression().fit(xtrain,ytrain)
pred=lr.predict(xtrain)
print(classification_report(ytrain,pred))
```

	precision	recall	f1-score	support
0	0.72	0.12	0.21	8748
1	0.63	0.97	0.76	13264
accuracy			0.63	22012
macro avg	0.67	0.55	0.48	22012
weighted avg	0.66	0.63	0.54	22012

Final Model (Random Forest)

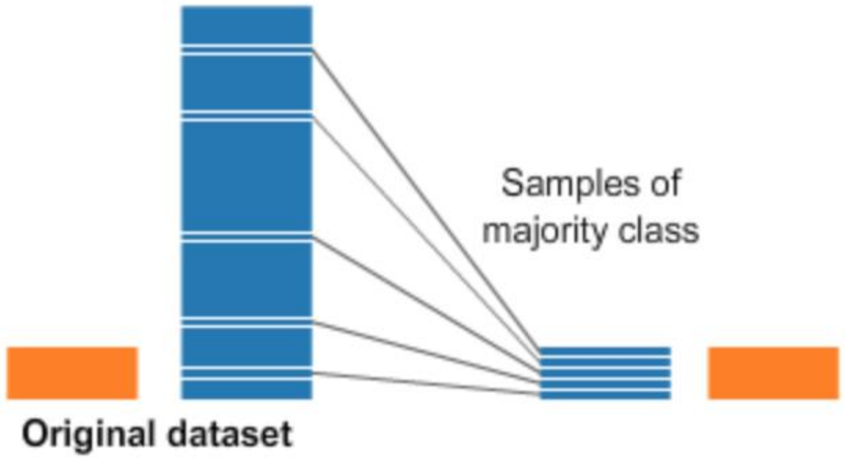
```
rf=RandomForestClassifier(max_depth=5,random_state=1).fit(xtrain,ytrain)
pred=rf.predict(xtrain)
print('Classification report for train set:\n\n',classification_report(ytrain,pred))
```

Classification report for train set:

	precision	recall	f1-score	support
0	0.70	0.59	0.64	8748
1	0.75	0.83	0.79	13264
accuracy			0.74	22012
macro avg	0.73	0.71	0.71	22012
weighted avg	0.73	0.74	0.73	22012

What could have been done better?

Under sampling of positive class in classification problem



Takeaways and Conclusions

Section A

- Restaurants to promote online orders
- Invest in seating to discourage table booking
- Enable delivery and provide variety in desert and dining

Section B

- Customers booking tables order online
- High rated and delivery restaurants get online orders
- Buffet restaurants, pubs, bars get offline orders

Future Steps

- Explanation of predictions using libraries like Lime
- Model deployment using packages like Streamlit
- Creation of dashboards for clients in Tableau or Power BI

Thank you.